RabbitCore RCM2000

C-Programmable Module

Getting Started Manual

019–0080 • 050501–G

RabbitCore RCM2000 Getting Started Manual

Part Number 019-0080 • 050501–G • Printed in U.S.A.

©2001–2005 Z-World Inc. • All rights reserved.

Z-World reserves the right to make changes and
improvements to its products without providing notice.

Trademarks

Rabbit is a registered trademark of Rabbit Semiconductor.

Rabbit 2000 and RabbitCore are trademarks of Rabbit Semiconductor.

Z-World is a registered trademark of Z-World Inc.

Dynamic C is a registered trademark of Z-World Inc.

Z-World, Inc.

2900 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-3737
Fax: (530) 757-3792

www.zworld.com

Rabbit Semiconductor

2932 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-8400
Fax: (530) 757-8402

www.rabbitsemiconductor.com

TABLE OF CONTENTS

Chapter 1. Overview 1

1.1 RCM2000 Description 1
1.2 Physical and Electrical Specifications 2
1.3 Development Software 3
1.4 How to Use This Manual 4
1.4.1 Additional Product Information 4
1.4.2 Additional Reference Information 4
1.4.3 Using Online Documentation 5

Chapter 2. Getting Started 7

2.1 Development Kit Contents 7
2.2 Overview of the Prototyping Board 8
2.3 Connections 9
2.4 Run a Sample Program 12
2.4.1 Troubleshooting 12
2.5 Where Do I Go From Here? 13
2.5.1 Technical Support 13

Chapter 3. Software Installation & Overview 15

3.1 An Overview of Dynamic C 15
3.2 Installing Dynamic C 17
3.3 Sample Programs 18
3.3.1 Running Sample Program FLASHLED.C 19
3.3.1.1 Single-Stepping 20
3.3.1.2 Watch Expressions 20
3.3.1.3 Break Point 20
3.3.1.4 Editing the Program 21
3.3.1.5 Watching Variables Dynamically 21
3.3.1.6 Summary of Features 21
3.3.1.7 Cooperative Multitasking 22
3.3.1.8 Advantages of Cooperative Multitasking 24
3.3.2 Getting to Know the RCM2000 25
3.3.3 Serial Communication 28
3.4 Upgrading Dynamic C 29
3.4.1 Add-On Modules 29

Notice to Users 31

Index 33

Schematics 35

1. OVERVIEW

The RCM2000 series of RabbitCore modules is an advanced line of modules that incorporates the powerful Rabbit® 2000 microprocessor, flash memory, and static RAM, all on a PCB not much larger than the size of a business card.

Throughout this manual, the term RCM2000 refers to the complete series of RCM2000 RabbitCore modules unless other production models are referred to specifically.

The RCM2000 modules are designed for use on a motherboard that supplies power and interfaces with real-world I/O devices. Up to 40 pins of I/O and four serial ports are available for system interfacing.

To accommodate a variety of user and production needs, the RCM2000 family includes versions with varying amounts of onboard memory. All modules within the family are pin-for-pin compatible and may be installed or swapped in a matter of minutes.

1.1 RCM2000 Description

There are three production models in the RCM2000 series. Their standard features are summarized in Table 1.

Table 1. RCM2000 Features

Model	Features
RCM2000	Full-featured RCM2000 module with 25.8 MHz clock, 256K flash memory, and 512K SRAM
RCM2010	RCM2000 with 25.8 MHz clock and 128K SRAM
RCM2020	RCM2000 with 18.432 MHz clock and 128K SRAM

The RCM2020 is the version that is included in the Development Kit.

1.2 Physical and Electrical Specifications

Table 2 lists the basic specifications for all models in the RCM2000 series.

Table 2. RCM2000 Specifications

Specification	Data
Power Supply	4.75 – 5.25 V DC (120 mA at 25.8048 MHz clock speed)
Size	1.90 × 2.30 × 0.55 inches (48.3 × 58.4 × 14 mm)
Environmental	–40°C to 70°C, 5–95% humidity, noncondensing

NOTE: For complete product specifications, see Appendix A in the ***RabbitCore RCM2000 User's Manual***.

The RCM2000 modules have two 40-pin headers to which cables can be connected, or which can be plugged into matching sockets on a production device. The pinouts for these connectors are shown in Figure 1 below.

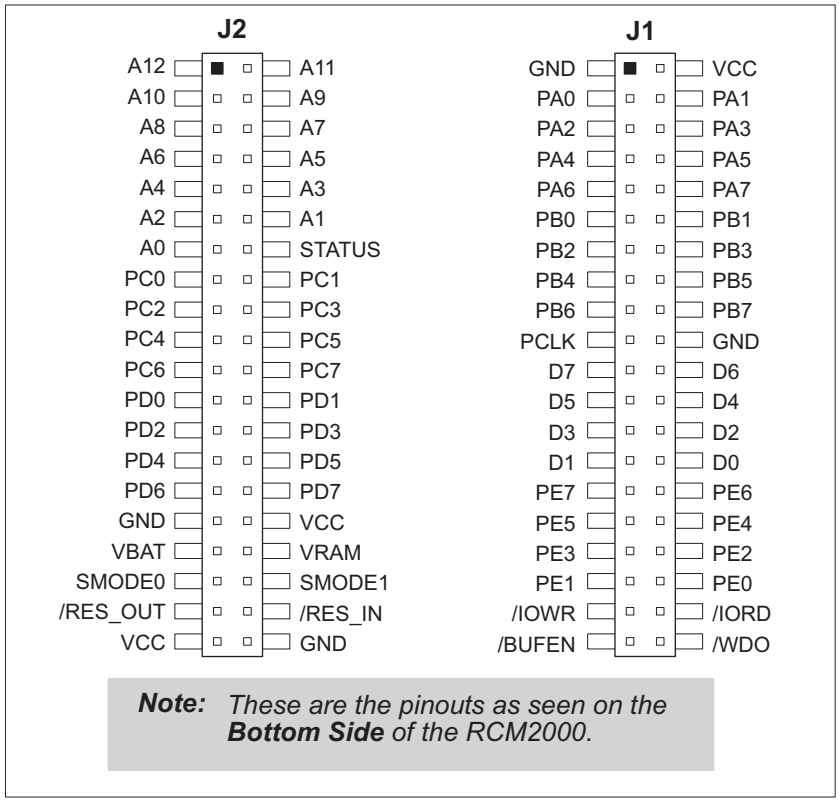


Figure 1. RCM2000 Connector Pinout

1.3 Development Software

The RCM2000 modules use the Dynamic C development environment for rapid creation and debugging of runtime applications. Dynamic C provides a complete development environment with integrated editor, compiler and source-level debugger. It interfaces directly with the target system, eliminating the need for complex and unreliable in-circuit emulators.

Dynamic C must be installed on a Windows workstation with at least one free serial (COM) port for communication with the target system. See Chapter 3., "Software Installation & Overview," for complete information on installing Dynamic C.

1.4 How to Use This Manual

This ***Getting Started*** manual is intended to give users a quick but solid start with RCM2000 modules. It does not contain detailed information on the module hardware capabilities or the Dynamic C development environment. Most users will want more detailed information on some or all of these topics in order to put the RCM2000 to effective use.

1.4.1 Additional Product Information

Detailed information about the RCM2000 is provided in the ***RabbitCore RCM2000 User's Manual***, which is available on the accompanying CD-ROM in both HTML and Adobe PDF format.

Some advanced users may choose to skip the rest of this introductory manual and proceed directly with the detailed hardware and software information in the ***User's Manual***.

> **TIP:** We recommend that anyone not thoroughly familiar with Z-World controllers at least read through the rest of this manual to gain the necessary familiarity to make use of the more advanced information.

1.4.2 Additional Reference Information

In addition to the product-specific information contained in the ***RabbitCore RCM2000 User's Manual***, several higher level reference manuals are provided in HTML and PDF form on the accompanying CD-ROM. Advanced users will find these references valuable in developing systems based on the RCM2000 modules:

- ***Dynamic C User's Manual***
- ***Rabbit 2000 Microprocessor User's Manual***

1.4.3 Using Online Documentation

We provide the bulk of our user and reference documentation in two electronic formats, HTML and Adobe PDF. We do this for several reasons.

We believe that providing all users with our complete library of product and reference manuals is a useful convenience. However, printed manuals are expensive to print, stock, and ship. Rather than include and charge for manuals that every user may not want, or provide only product-specific manuals, we chose to provide our complete documentation and reference library in electronic form with every Development Kit and with our Dynamic C development environment.

Finding Online Documents

The online documentation is installed along with Dynamic C, and an icon for the documentation menu is placed on the workstation's desktop. Double-click this icon to reach the menu. If the icon is missing, create a new desktop icon that points to **default.htm** in the **docs** folder, found in the Dynamic C installation folder.

The latest versions of all documents are always available for free, unregistered download from our Web sites as well.

Printing Electronic Manuals

We recognize that many users prefer printed manuals for some uses. Users can easily print all or parts of those manuals provided in electronic form. The following guidelines may be helpful:

- Print from the Adobe PDF versions of the files, not the HTML versions.

  NOTE: The most current version of Adobe Acrobat Reader can always be downloaded from Adobe's web site at **http://www.adobe.com**. We recommend that you use version 4.0 or later.

- Print only the sections you will need to refer to often.
- Print manuals overnight, when appropriate, to keep from tying up shared resources during the work day.
- If your printer supports duplex printing, print pages double-sided to save paper and increase convenience.

  NOTE: If you do not have a suitable printer or do not want to print the manual yourself, most retail copy shops (e.g., Kinkos, AlphaGraphics, etc.) will print the manual from the PDF file and bind it for a reasonable charge—about what we would have to charge for a printed and bound manual.

2. GETTING STARTED

This chapter describes the RCM2000 hardware in more detail, and explains how to set up the accompanying Prototyping Board.

NOTE: This chapter (and this manual) assume that you have the RabbitCore RCM2000 Development Kit. If you purchased an RCM2000 module by itself, you will have to adapt the information in this chapter and elsewhere to your test and development setup.

2.1 Development Kit Contents

The RCM2000 Development Kit contains the following items:

- RCM2020 module with 256K flash memory and 128K SRAM.
- RCM2000 Prototyping Board with accessory hardware and components.
- Wall transformer power supply, 12 V DC, 1 A (included only with Development Kits sold for the North American market. Overseas users will need a power supply compatible with their local mains power).
- 10-pin header to DE9 programming cable with integrated level-matching circuitry.
- ***Dynamic C*** CD-ROM, with complete product documentation on disk.
- This ***Getting Started*** manual.
- Registration card.

2.2 Overview of the Prototyping Board

The Prototyping Board included in the Development Kit makes it easy to connect an RCM2000 module to a power supply and a PC workstation for development. It also provides an array of basic I/O peripherals (switches and LEDs), as well as a prototyping area for more advanced hardware development.

For the most basic level of evaluation and development, the Prototyping Board can be used without modification.

As you progress to more sophisticated experimentation and hardware development, modifications and additions can be made to the board without modifying or damaging the RCM2000 itself.

The Prototyping Board is shown in Figure 2 below, with its main features identified.

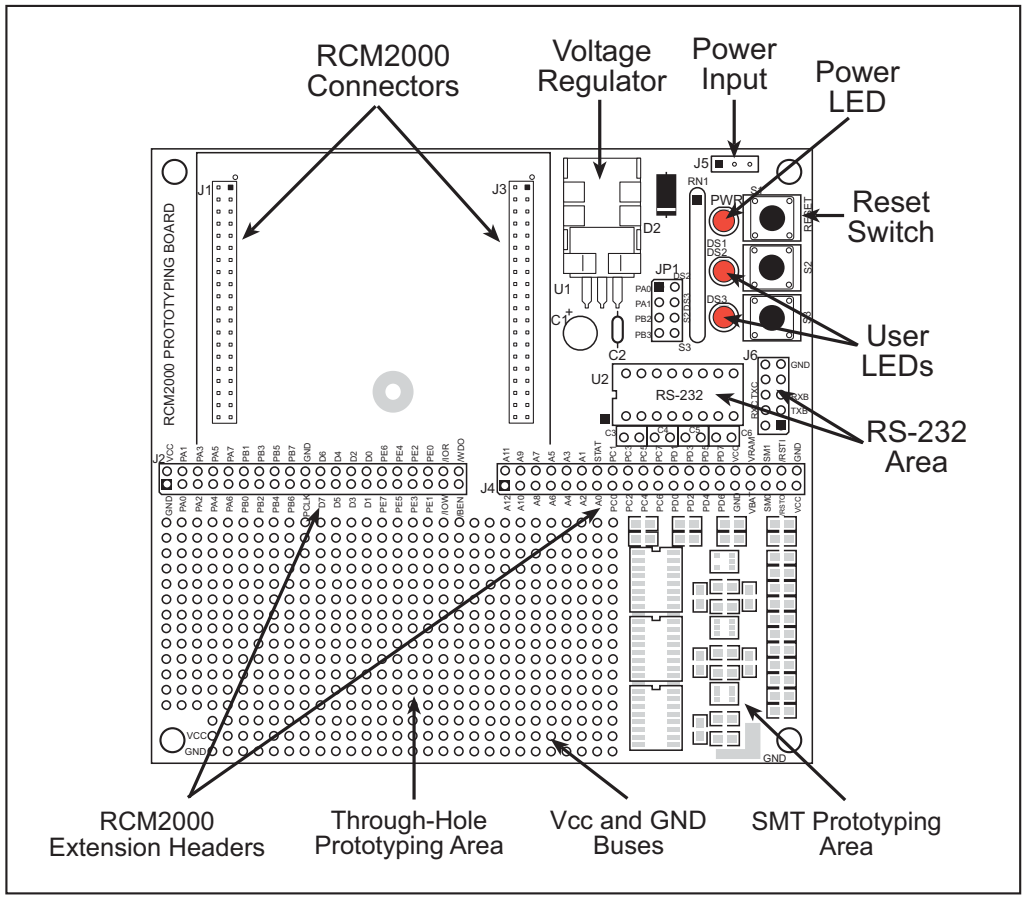


Figure 2. RCM2000 Prototyping Board

2.3 Connections

1. Attach RCM2000 to Prototyping Board

Turn the RCM2000 so that the Rabbit 2000 microprocessor is facing as shown below. Plug RCM2000 headers J1 and J2 on the bottom side of the RCM2000 into the sockets of headers J1 and J3 on the Prototyping Board.

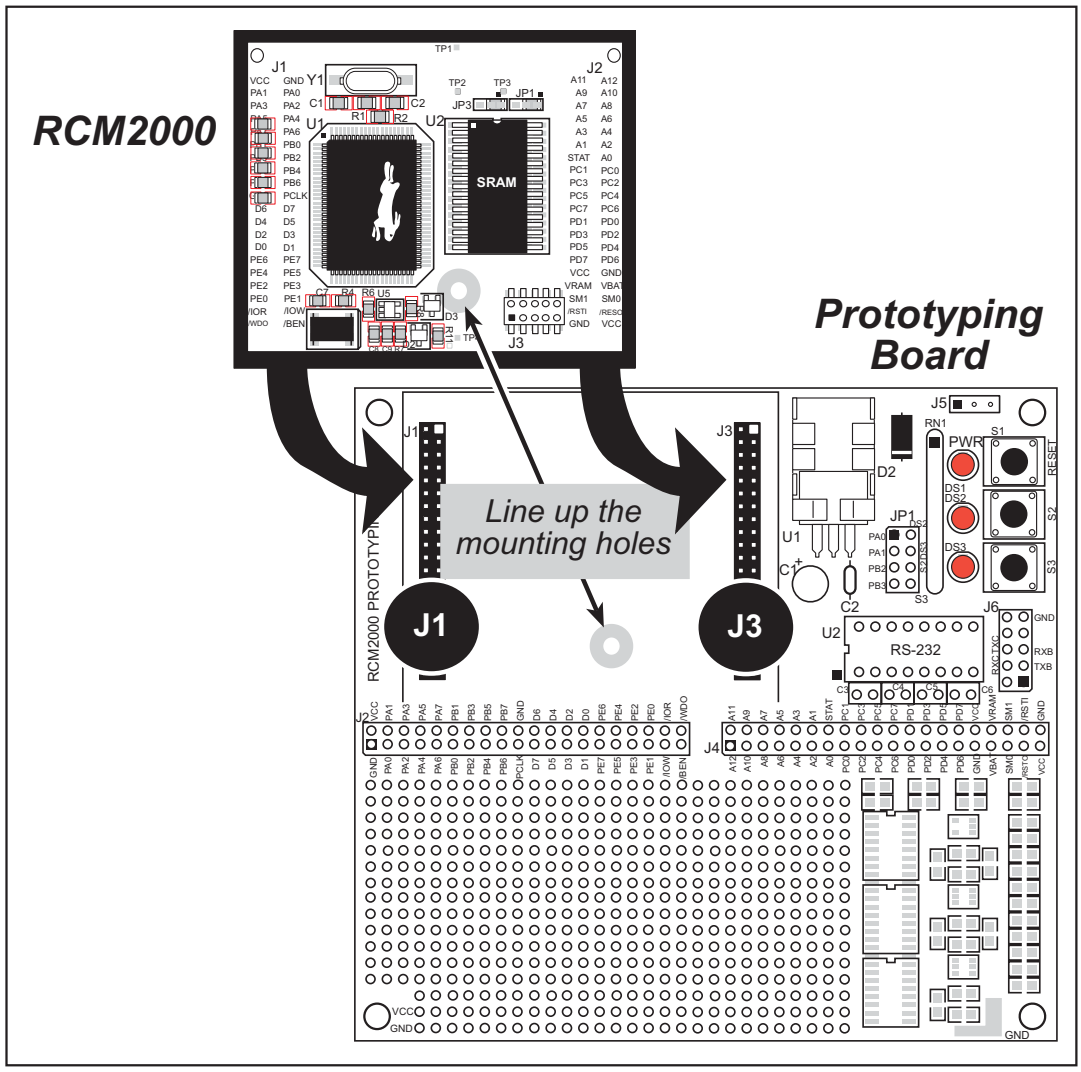


Figure 3. Attaching RCM2000 to Prototyping Board

NOTE: It is important that you line up the pins on the RCM2000 headers J1 and J2 exactly with the corresponding pins of header sockets J1 and J3 on the Prototyping Board. The header pins may become bent or damaged if the pin alignment is offset, and the RCM2000 will not work.

2. Connect RCM2000 to PC

Connect the 10-pin connector of the programming cable labeled **PROG** to header J3 on the RCM2000 module as shown in Figure 4 below. Be sure to orient the red edge of the cable towards pin 1 of the connector. (Do not use the **DIAG** connector, which is used for a normal serial connection.)

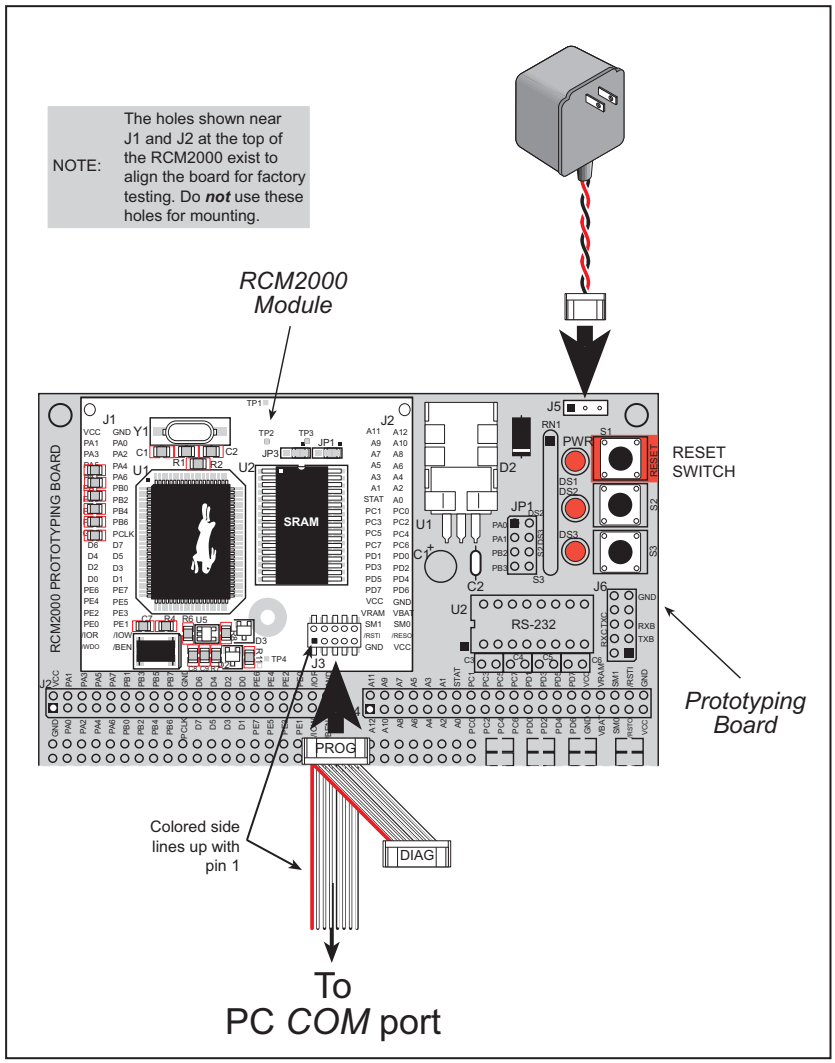

NOTE:
The holes shown near J1 and J2 at the top of the RCM2000 exist to align the board for factory testing. Do *not* use these holes for mounting.
RCM2000 Module
RESET SWITCH
Prototyping Board
PROG
DIAG
Colored side lines up with pin 1
To PC *COM* port
RCM2000 PROTOTYPING BOARD
SRAM
RS-232

Figure 4. RCM2000 Power and Programming Connections

NOTE: Some PCs now come equipped only with a USB port. It may be possible to use an RS-232/USB converter with the programming cable supplied with your RCM2000 module. An RS-232/USB converter is available through the Z-World Web store.

3. Power Supply Connections

Hook up the connector from the wall transformer to header J5 on the Prototyping Board as shown in Figure 4. The orientation of this connector is not important since the VIN (positive) voltage is the middle pin, and GND is available on both ends of the three-pin header J5.

Plug in the wall transformer. The power LED on the Prototyping Board should light up. The RCM2000 and the Prototyping Board are now ready to be used.

NOTE: A RESET button is provided on the Prototyping Board to allow a hardware reset.

2.4 Run a Sample Program

If you already have Dynamic C installed, you are now ready to test your programming connections by running a sample program.

If you are using a USB port to connect your computer to the RCM2000 module, choose **Options > Project Options** and select "Use USB to Serial Converter" under the **Communications** tab.

Find the file **PONG.C**, which is in the Dynamic C **SAMPLES** folder. To run the program, open it with the **File** menu (if it is not still open), then compile and run it by pressing **F9** or by selecting **Run** in the **Run** menu. The **STDIO** window will open and will display a small square bouncing around in a box.

2.4.1 Troubleshooting

If Dynamic C appears to compile the BIOS successfully, but you then receive a communication error message when you compile and load the sample program, it is possible that your PC cannot handle the higher program-loading baud rate. Try changing the maximum download rate to a slower baud rate as follows.

- Locate the **Serial Options** dialog in the Dynamic C **Options > Project Options > Communications** menu. Select a slower Max download baud rate.

If a program compiles and loads, but then loses target communication before you can begin debugging, it is possible that your PC cannot handle the default debugging baud rate. Try lowering the debugging baud rate as follows.

- Locate the **Serial Options** dialog in the Dynamic C **Options > Project Options > Communications** menu. Choose a lower debug baud rate.

If there are any other problems:

- Check to make sure you are using the **PROG** connector, not the **DIAG** connector, on the programming cable.
- Check both ends of the programming cable to ensure that they are firmly plugged into the PC and the programming port on the RCM2000.
- Ensure that the RCM2000 module is firmly and correctly installed in its connectors on the Prototyping Board.
- Select a different COM port within Dynamic C. From the **Options** menu, select **Project Options**, then select **Communications**. Select another COM port from the list, then click OK. Press **<Ctrl-Y>** to force Dynamic C to recompile the BIOS. If Dynamic C still reports it is unable to locate the target system, repeat the above steps until you locate the active COM port.

2.5 Where Do I Go From Here?

If everything appears to be working, we recommend the following sequence of action:

1. Run all of the sample programs described in Chapter 3 to get a basic familiarity with Dynamic C and the RCM2000's capabilities.
2. For further development, refer to the ***RabbitCore RCM2000 User's Manual*** for details of the module's hardware and software components.

   A documentation icon should have been installed on your workstation's desktop; click on it to reach the documentation menu. You can create a new desktop icon that points to **default.htm** in the **docs** folder in the Dynamic C installation folder.
3. For advanced development topics, refer to the ***Dynamic C User's Manual***, also in the online documentation set.

2.5.1 Technical Support

> **NOTE:** If you purchased your RCM2000 through a distributor or through a Z-World or Rabbit Semiconductor partner, contact the distributor or partner first for technical support.

If there are any problems at this point:

- Use the Dynamic C **Help** menu to get further assistance with Dynamic C.
- Check the Z-World/Rabbit Semiconductor Technical Bulletin Board at www.zworld.com/support/bb/.
- Use the Technical Support e-mail form at www.zworld.com/support/questionSubmit.shtml.

3. SOFTWARE INSTALLATION & OVERVIEW

To develop and debug programs for the RCM2000 (and for all other Z-World and Rabbit Semiconductor hardware), you must install and use Dynamic C. Dynamic C is an integrated development system for writing embedded software. It runs on an IBM-compatible PC and is designed for use with Z-World single-board computers and other single-board computers based on the Rabbit microprocessor. Chapter 3 provides the libraries, function calls, and sample programs related to the RCM2000.

3.1 An Overview of Dynamic C

Dynamic C has been in use worldwide since 1989. It is specially designed for programming embedded systems, and features quick compile and interactive debugging. A complete reference guide to Dynamic C is contained in the ***Dynamic C User's Manual***.

You have a choice of doing your software development in the flash memory or in the SRAM included on the RCM2000. The flash memory and SRAM options are selected with the **Options > Project Options > Compiler** menu.

The advantage of working in RAM is to save wear on the flash memory, which is limited to about 100,000 write cycles. The disadvantage is that the code and data might not both fit in RAM.

> **NOTE:** An application can be developed in RAM, but cannot run standalone from RAM after the programming cable is disconnected. All standalone applications can only run from flash memory.

> **NOTE:** Do not depend on the flash memory sector size or type. Due to the volatility of the flash memory market, the RCM2000 and Dynamic C were designed to accommodate flash devices with various sector sizes.

Developing software with Dynamic C is simple. Users can write, compile, and test C and assembly code without leaving the Dynamic C development environment. Debugging occurs while the application runs on the target. Alternatively, users can compile a program to an image file for later loading. Dynamic C runs on PCs under Windows 95, 98, 2000, NT, Me, and XP. Programs can be downloaded at baud rates of up to 460,800 bps after the program compiles.

Dynamic C has a number of standard features.

- Full-feature source and/or assembly-level debugger, no in-circuit emulator required.
- Royalty-free TCP/IP stack with source code and most common protocols.
- Hundreds of functions in source-code libraries and sample programs:
  - Exceptionally fast support for floating-point arithmetic and transcendental functions.
  - RS-232 and RS-485 serial communication.
  - Analog and digital I/O drivers.
  - I^2C, SPI, GPS, file system.
  - LCD display and keypad drivers.
- Powerful language extensions for cooperative or preemptive multitasking
- Loader utility program to load binary images into Z-World targets in the absence of Dynamic C.
- Provision for customers to create their own source code libraries and augment on-line help by creating "function description" block comments using a special format for library functions.
- Standard debugging features:
  - Breakpoints—Set breakpoints that can disable interrupts.
  - Single-stepping—Step into or over functions at a source or machine code level, µC/OS-II aware.
  - Code disassembly—The disassembly window displays addresses, opcodes, mnemonics, and machine cycle times. Switch between debugging at machine-code level and source-code level by simply opening or closing the disassembly window.
  - Watch expressions—Watch expressions are compiled when defined, so complex expressions including function calls may be placed into watch expressions. Watch expressions can be updated with or without stopping program execution.
  - Register window—All processor registers and flags are displayed. The contents of general registers may be modified in the window by the user.
  - Stack window—shows the contents of the top of the stack.
  - Hex memory dump—displays the contents of memory at any address.
  - **STDIO** window—`printf` outputs to this window and keyboard input on the host PC can be detected for debugging purposes. `printf` output may also be sent to a serial port or file.

3.2 Installing Dynamic C

Insert the Dynamic C CD from the Development Kit in your PC's CD-ROM drive. If the installation does not auto-start, run the **setup.exe** program in the root directory of the Dynamic C CD. Install any Dynamic C modules after you install Dynamic C.

Dynamic C has two components that can be installed together or separately. One component is Dynamic C itself, with the development environment, support files and libraries. The other component is the documentation library in HTML and PDF formats, which may be left uninstalled to save hard drive space or installed elsewhere (on a separate or network drive, for example).

The installation type is selected in the installation menu. The options are:

- **Typical Installation —** Both Dynamic C and the documentation library will be installed in the specified folder (default).
- **Compact Installation —** Only Dynamic C will be installed.
- **Custom Installation —** You will be allowed to choose which components are installed. This choice is useful to install or reinstall just the documentation.

3.3 Sample Programs

To help familiarize you with the RCM2000 modules, Dynamic C includes several sample programs in the Dynamic C **SAMPLES\RCM2000** directory. Loading, executing and studying these programs will give you a solid hands-on overview of the RCM2000's capabilities, as well as a quick start with Dynamic C as an application development tool. These programs are intended to serve as tutorials, but then can also be used as starting points or building blocks for your own applications.

> **NOTE:** It is assumed in this section that you have at least an elementary grasp of ANSI C. If you do not, see the introductory pages of the ***Dynamic C User's Manual*** for a suggested reading list.

Each sample program has comments that describe the purpose and function of the program.

Before running any of these sample program, make sure that your RCM2000 is connected to the Prototyping Board and to your PC as described in Section 2.3, "Connections."

To run a sample program, open it with the **File** menu (if it is not already open), then compile and run it by pressing **F9** or by selecting **Run** in the **Run** menu.

More complete information on Dynamic C is provided in the ***Dynamic C User's Manual***.

3.3.1 Running Sample Program FLASHLED.C

This sample program will be used to illustrate some of the functions of Dynamic C.

First, open the file **FLASHLED.C**, which is in the **SAMPLES/RCM2000** folder. The program will appear in a window, as shown in Figure 5 below (minus some comments). Use the mouse to place the cursor on the function name **WrPortI** in the program and type **<Ctrl-H>**. This will bring up a documentation box for the function **WrPortI**. In general, you can do this with all functions in Dynamic C libraries, including libraries you write yourself. Close the documentation box and continue.

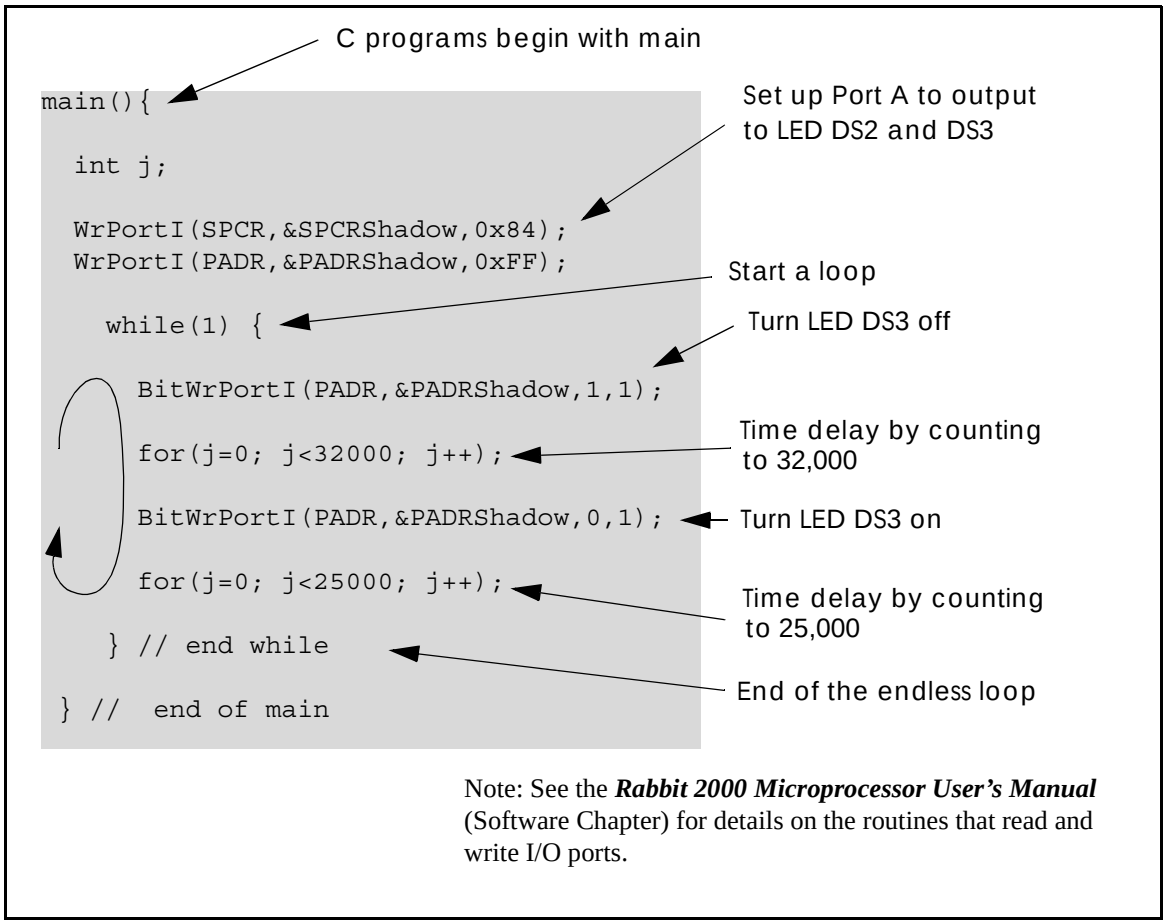

C programs begin with main
main(){
int j;
WrPortI(SPCR,&SPCRShadow,0x84);
WrPortI(PADR,&PADRShadow,0xFF);
while(1) {
BitWrPortI(PADR,&PADRShadow,1,1);
for(j=0; j<32000; j++);
BitWrPortI(PADR,&PADRShadow,0,1);
for(j=0; j<25000; j++);
} // end while
} // end of main
Set up Port A to output to LED DS2 and DS3
Start a loop
Turn LED DS3 off
Time delay by counting to 32,000
Turn LED DS3 on
Time delay by counting to 25,000
End of the endless loop
Note: See the Rabbit 2000 Microprocessor User's Manual (Software Chapter) for details on the routines that read and write I/O ports.

Figure 5. Sample Program FLASHLED.C

To run the program **FLASHLED.C**, open it with the **File** menu (if it is not already open), then compile and run it by pressing **F9** or by selecting **Run** in the **Run** menu. The LED on the Prototyping Board should start flashing if everything went well. If this doesn't work review the following points.

- The target should be ready, which is indicated by the message "BIOS successfully compiled..." If you did not receive this message or you get a communication error, recompile the BIOS by typing **<Ctrl-Y>** or select **Recompile BIOS** from the **Compile** menu.

- A message reports “No Rabbit Processor Detected” in cases where the RCM2000 and the Prototyping Board are not connected together, the wall transformer is not connected, or is not plugged in. (The red power LED lights whenever power is connected.)
- The programming cable must be connected to the RCM2000. (The colored wire on the programming cable is closest to pin 1 on header J3 on the RCM2000, as shown in Figure 4.) The other end of the programming cable must be connected to the PC serial port. The COM port specified in the Dynamic C **Options** menu must be the same as the one the programming cable is connected to.
- To check if you have the correct serial port, select **Compile**, then **Compile BIOS**, or type **<Ctrl-Y>**. If the “BIOS successfully compiled …” message does not display, try a different serial port using the Dynamic C **Options** menu until you find the serial port you are plugged into. Don’t change anything in this menu except the COM number. The baud rate should be 115,200 bps and the stop bits should be 1.

3.3.1.1 Single-Stepping

Compile or re-compile **FLASHLED.C** by clicking the **Compile** button on the task bar. The program will compile and the screen will come up with a highlighted character (green) at the first executable statement of the program. Use the **F8** key to single-step. Each time the **F8** key is pressed, the cursor will advance one statement. When you get to the **for(j=0, j< ...** statement, it becomes impractical to single-step further because you would have to press **F8** thousands of times. We will use this statement to illustrate watch expressions.

3.3.1.2 Watch Expressions

Type **<Ctrl-W>** or chose **Add/Del Watch Expression** in the **Inspect** menu. A box will come up. Type the lower case letter j and click on *add to top* and *close*. Now continue single-stepping with **F8**. Each time you step, the watch expression (**j**) will be evaluated and printed in the watch window. Note how the value of **j** advances when the statement **j++** is executed.

3.3.1.3 Break Point

Move the cursor to the start of the statement:

```
for(j=0; j<25000; j++);
```

To set a break point on this statement, type **F2** or select **Toggle Breakpoint** from the **Run** menu. A red highlight will appear on the first character of the statement. To get the program running at full speed, type **F9** or select **Run** on the **Run** menu. The program will advance until it hits the break point. Then the break point will start flashing and show both red and green colors. Note that LED DS3 is now solidly turned on. This is because we have passed the statement turning on LED DS3. Note that **j** in the watch window has the value 32000. This is because the loop above terminated when **j** reached 32000.

To remove the break point, type **F2** or select **Toggle Breakpoint** on the **Run** menu. To continue program execution, type **F9** or select **Run** from the **Run** menu. Now the LED should be flashing again since the program is running at full speed.

You can set break points while the program is running by positioning the cursor to a statement and using the **F2** key. If the execution thread hits the break point, a break point will take place.You can toggle the break point off with the **F2** key and continue execution with the **F9** key. Try this a few times to get the feel of things.

3.3.1.4 Editing the Program

Click on the **Edit** box on the task bar. This will set Dynamic C into the edit mode so that you can change the program. Use the **Save as** choice on the **File** menu to save the file with a new name so as not to change the demo program. Save the file as **MYTEST.C**. Now change the number 25000 in the **for (..** statement to 10000. Then use the **F9** key to recompile and run the program. The LED will start flashing, but it will flash much faster than before because you have changed the loop counter terminal value from 25000 to 10000.

3.3.1.5 Watching Variables Dynamically

Go back to edit mode (select edit) and load the program **FLASHLED2.C** using the **File** menu **Open** command. This program is the same as the first program, except that a variable **k** has been added along with a statement to increment **k** each time around the endless loop. The statement:

```
runwatch();
```

has been added. This is a debugging statement that makes it possible to view variables while the program is running.

Use the **F9** key to compile and run **FLASHLED2.C**. Now type **<Ctrl-W>** to open the watch window and add the watch expression **k** to the top of the list of watch expressions. Now type **<Ctrl-U>**. Each time you type **<Ctrl-U>**, you will see the current value of **k**, which is incrementing about 5 times a second.

As an experiment, add another expression to the watch window:

```
k*5
```

Then type **<ctrl-U>** several times to observe the watch expressions **k** and **k*5**.

3.3.1.6 Summary of Features

So far you have practiced using the following features of Dynamic C.

- Loading, compiling and running a program. When you load a program it appears in an edit window. You can compile by selecting **Compile** on the task bar or from the **Compile** menu. When you compile the program, it is compiled into machine language and downloaded to the target over the serial port. The execution proceeds to the first statement of main where it pauses, waiting for you to command the program to run, which you can do with the **F9** key or by selecting **Run** on the **Run** menu. If want to compile and start the program running with one keystroke, use **F9**, the run command. If the program is not already compiled, the run command will compile it first.
- Single-stepping. This is done with the **F8** key. The **F7** key can also be used for single-stepping. If the **F7** key is used, then descent into subroutines will take place. With the **F8** key the subroutine is executed at full speed when the statement that calls it is stepped over.

- Setting break points. The **F2** key is used to turn on or turn off (toggle) a break point at the cursor position if the program has already been compiled. You can set a break point if the program is paused at a break point. You can also set a break point in a program that is running at full speed. This will cause the program to break if the execution thread hits your break point.
- Watch expressions. A watch expression is a C expression that is evaluated on command in the watch window. An expression is basically any type of C formula that can include operators, variables and function calls, but not statements that require multiple lines such as *for* or *switch*. You can have a list of watch expressions in the watch window. If you are single-stepping, then they are all evaluated on each step. You can also command the watch expression to be evaluated by using the **<Ctrl-U>** command. When a watch expression is evaluated at a break point, it is evaluated as if the statement was at the beginning of the function where you are single-stepping. If your program is running you can also evaluate watch expressions with a **<Ctrl-U>** if your program has a `runwatch()` command that is frequently executed. In this case, only expressions involving global variables can be evaluated, and the expression is evaluated as if it were in a separate function with no local variables.

3.3.1.7 Cooperative Multitasking

Cooperative multitasking is a convenient way to perform several different tasks at the same time. An example would be to step a machine through a sequence of steps and at the same time independently carry on a dialog with the operator via a human interface. Cooperative multitasking differs from another approach called preemptive multitasking. Dynamic C supports both types of multitasking. In cooperative multitasking each separate task voluntarily surrenders its compute time when it does not need to perform any more activity immediately. In preemptive multitasking control is forcibly removed from the task via an interrupt.

Dynamic C has language extensions to support multitasking. The major C constructs are called *costatements, cofunctions,* and *slicing*. These are described more completely in the ***Dynamic C User's Manual***. The example below, sample program `FLASHLEDS2.C`, uses costatements. A costatement is a way to perform a sequence of operations that involve pauses or waits for some external event to take place. A complete description of costatements is in the ***Dynamic C User's Manual***. The `FLASHLEDS2.C` sample program has two independent tasks. The first task flashes LED DS2 2.5 times a second. The second task flashes DS3 every 1.5 seconds.

```
#define DS2 0           // predefine for LED DS2
#define DS3 1           // predefine for LED DS3

//  This cofunction flashes LED on for ontime, then off for offtime
cofunc flashled[4](int led, int ontime, int offtime) {
    for(;;) {
        waitfor(DelayMs(ontime));                       // on delay
        WrPortI(PADR,&PADRShadow,(1<<led)|PADR);        // turn LED off
        waitfor(DelayMs(offtime);                       // off delay
        WrPortI(PADR,&PADRShadow,(1<<led)^0xff&PADR);   // turn LED on
    }
}

main {
    // Initialize ports
    WrPortI(SPCR,&SPCRShadow,0x84);     // Set Port A all outputs, LEDs on
    WrPortI(PEFR,&PEFRShadow,0x00);     // Set Port E normal I/O
    WrPortI(PEDDR,&PEDDRShadow,0x01);  // Set Port E bits 7…1 input, 0 output
    WrPortI(PECR,&PECRShadow,0x00);     // Set transfer clock as pclk/2

    for(;;) {                           // run forever
        costate {                       // start costatement
            wfd {                 // use wfd (waitfordone) with cofunctions
              flashled[0](DS2,200,200); // flash DS2 on 200 ms, off 200 ms
              flashled[1](DS3,1000,500);// flash DS3 on 1000 ms, off 500 ms
            }
        }                                // end costatement
    }                                    // end for loop
}                                         // end of main, never come here
```

The flashing of the LEDs is performed by the costatement. Costatements need to be executed regularly, often at least every 25 ms. To accomplish this, the costatements are enclosed in a **while** loop or a **for** loop. The term **while** loop is used as a handy way to describe a style of real-time programming in which most operations are done in one loop.

The costatement is executed on each pass through the big loop. When a **waitfor** or a **wfd** condition is encountered the first time, the current value of **MS_TIMER** is saved and then on each subsequent pass the saved value is compared to the current value. If a **waitfor** condition is not encountered, then a jump is made to the end of the costatement, and on the next pass of the loop, when the execution thread reaches the beginning of the costatement, execution passes directly to the **waitfor** statement. The costatement has the property that it can wait for long periods of time, but not use a lot of execution time. Each costatement is a little program with its own statement pointer that advances in response to conditions. On each pass through the big loop, as little as one statement in the costatement is executed, starting at the current position of the costatement's statement pointer. Consult the ***Dynamic C User's Manual*** for more details.

This program also illustrates a use for a shadow register. A shadow register is used to keep track of the contents of an I/O port that is write only—it can't be read back. If every time a write is made to the port the same bits are set in the shadow register, then the shadow register has the same data as the port register.

3.3.1.8 Advantages of Cooperative Multitasking

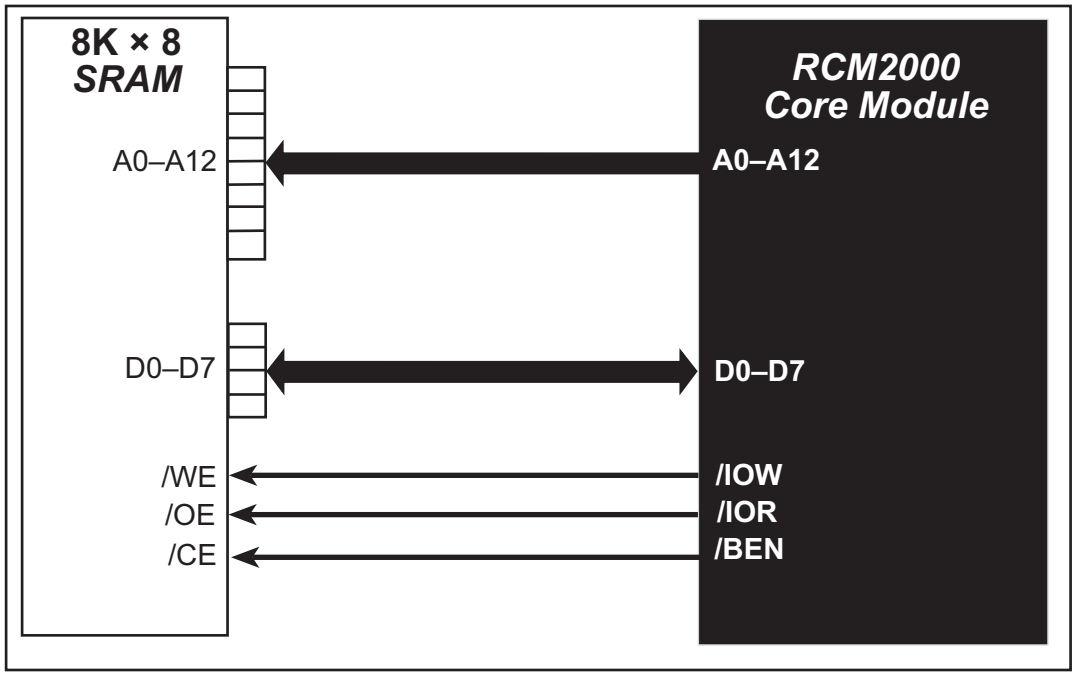
Cooperative multitasking, as implemented with language extensions, has the advantage of being intuitive. Unlike preemptive multitasking, variables can be shared between different tasks without having to take elaborate precautions. Sharing variables between tasks is the greatest cause of bugs in programs that use preemptive multitasking. It might seem that the biggest problem would be response time because of the big loop time becoming long as the program grows. Our solution for that is called slicing, which is further described in the ***Dynamic C User's Manual***.

3.3.2 Getting to Know the RCM2000

The following sample programs can be found in the **SAMPLES\RCM2000** folder.

- **EXTSRAM.C**—demonstrates the setup and simple addressing to an external SRAM. This program first maps the external SRAM to the I/O Bank 0 register with a maximum of 15 wait states, chip select strobe (which is ignored because of the circuitry), and allows writes. The first 256 bytes of SRAM are cleared and read back. Values are then written to the same area and are read back. The Dynamic C **STDIO** window will indicate if writes and reads did not occur

  Connect an external SRAM as shown below before you run this sample program.



- **FLASHLED.C**—repeatedly flashes LED DS3 on the Prototyping Board on and off. LED DS3 is controlled by Parallel Port A bit 1 (PA1).
- **FLASHLED2.C**—repeatedly flashes LED DS3 on the Prototyping Board on and off. LED DS3 is controlled by Parallel Port A bit 1 (PA1).

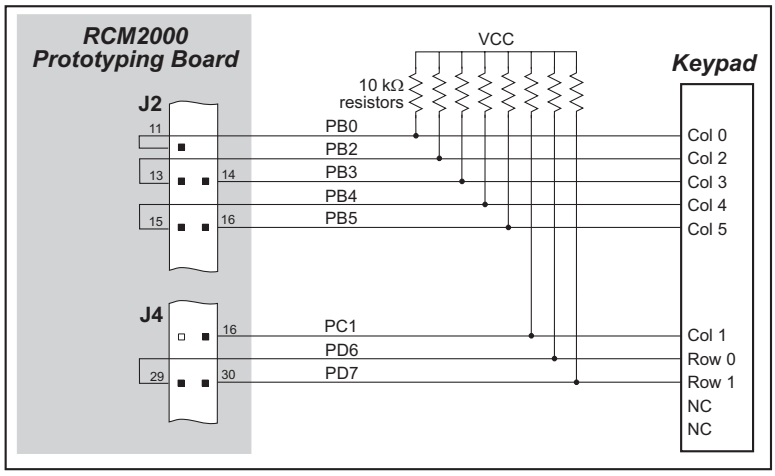
  This sample program also shows the use of the **runwatch()** function to allow Dynamic C to update watch expressions while running. The following steps explain how to do this.

  1. Add a watch expression for "k" in the **Inspect > Add Watch** dialog box.
  2. Click "Add" or "Add to top" so that it will be in the watch list permanently.
  3. Click **OK** to close the dialog box.
  4. Press **<Ctrl+U>** while the program is running. This will update the watch window

- **FLASHLEDS.C**—demonstrates the use of coding with assembly instructions, cofunctions, and costatements to flash LEDs DS2 and DS3 on the Prototyping Board on and off. LEDs DS2 and DS3 are controlled by Parallel Port A bit 0 (PA0) and Parallel Port A bit 1 (PA1).Once you have compile this program and it is running, LEDs DS2 and DS3 will flash on/off at different rates.
- **FLASHLEDS2.C**—demonstrates the use of cofunctions and costatements to flash LEDs DS2 and DS3 on the Prototyping Board on and off. LEDs DS2 and DS3 are controlled by Parallel Port A bit 0 (PA0) and Parallel Port A bit 1 (PA1).Once you have compile this program and it is running, LEDs DS2 and DS3 will flash on/off at different rates.
- **KEYLCD.C**—demonstrates a simple setup for a 2 × 6 keypad and a 2 × 20 LCD.

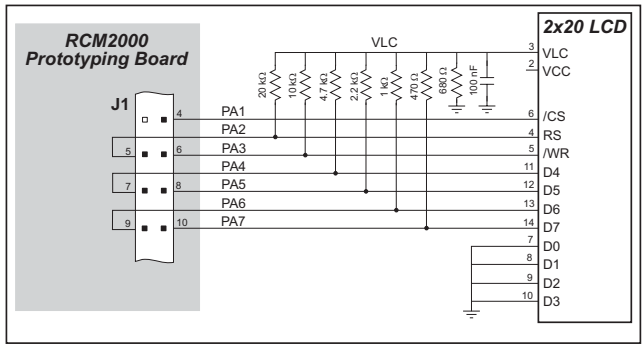
  Connect the keypad to Parallel Ports B, C, and D.

  PB0—Keypad Col 0
  PC1—Keypad Col 1
  PB2—Keypad Col 2
  PB3—Keypad Col 3
  PB4—Keypad Col 4
  PB5—Keypad Col 5
  PD6—Keypad Row 0
  PD7—Keypad Row 1



  Connect the LCD to Parallel Port A.

  PA0—backlight (if connected)
  PA1—LCD /CS
  PA2—LCD RS (High = Control, Low = Data) / LCD Contrast 0
  PA3—LCD /WR/ LCD Contrast 1
  PA4—LCD D4 / LCD Contrast 2
  PA5—LCD D5 / LCD Contrast 3
  PA6—LCD D6 / LCD Contrast 4
  PA7—LCD D7 / LCD Contrast 5

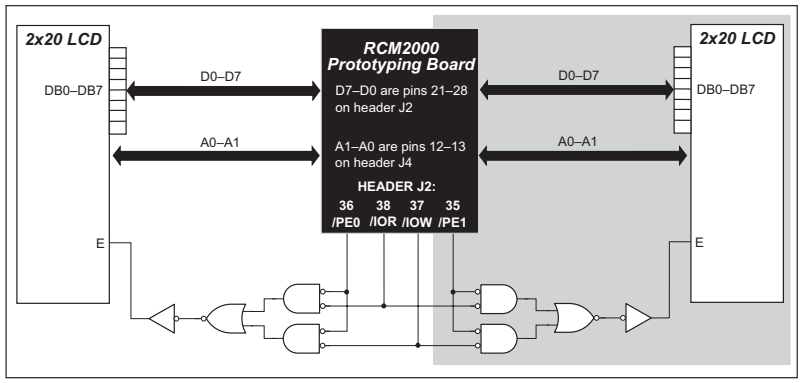


  Once the connections have been made and the sample program is running, the LCD will display two rows of 6 dots, each dot representing the corresponding key. When a key is pressed, the corresponding dot will become an asterisk.

- **LCD_DEMO.C**—demonstrates a simple setup for an LCD that uses the HD44780 controller or an equivalent.

  Connect the LCD to the RCM2000 address and data lines on the Prototyping Board.

  D0—DB0
  D1—DB1
  D2—DB2
  D3—DB3
  D4—DB4
  D5—DB5
  D6—DB6
  D7—DB7



  A0—RS (Register Select: 0 = command, 1 = data)
  A1—R/W (0=write, 1=read)
  *—E (normally low: latches on high-to-low transition)

- **SWTEST.C**—demonstrates the use of pushbutton switches S2 and S3 to toggle LEDs DS2 and DS3 on the Prototyping Board on and off.

  Parallel Port A bit 0 = LED DS2
  Parallel Port A bit 1 = LED DS3

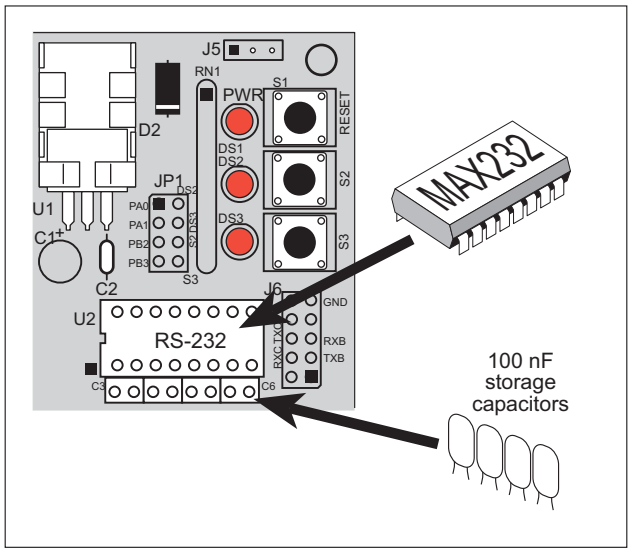
  Parallel Port B bit 2 = switch S2
  Parallel Port B bit 3 = switch S3

- **TOGGLELED.C**—demonstrates the use of costatements to detect switch presses using the press-and-release method of debouncing. As soon as the sample program starts running, LED DS3 on the Prototyping Board (which is controlled by PA1) starts flashing once per second. Press switch S2 on the Prototyping Board (which is connected to PB2) to toggle LED DS2 on the Prototyping Board (which is controlled by PA0). The pushbutton switch is debounced by the software.

3.3.3 Serial Communication

The following sample programs can be found in the **SAMPLES\RCM2000** folder.

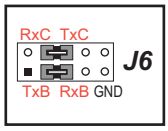
Two sample programs, **CORE_FLOW-CONTROL.C** and **CORE_PARITY.C**, are available to illustrate RS-232 communication. To run these sample programs, you will have to add an RS-232 transceiver such as the MAX232 at location U2 and four 100 nF charge-storage capacitors at C3–C6 on the Prototyping Board. Also install the 2 × 5 IDC header included with the Prototyping Board accessory parts at J6 to interface the RS-232 signals.



The diagram shows the connections.

- **CORE_FLOWCONTROL.C**—This program demonstrates hardware flow control by configuring Serial Port C (PC3/PC2) for CTS/RTS with serial data coming from TxB at 115,200 bps. One character at a time is received and is displayed in the **STDIO** window.

  To set up the Prototyping Board, you will need to tie PC4 and PC5 (TxB and RxB) together at header J4, and you will also tie PC2 and PC3 (TxC and RxC) together as shown in the diagram.

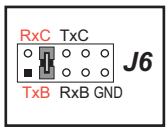
  

  A repeating triangular pattern should print out in the **STDIO** window. The program will periodically switch flow control on or off to demonstrate the effect of no flow control.

  Refer to the **serBflowcontrolOn()** function call in the ***Dynamic C Function Reference Manual*** for a general description on how to set up flow control lines.

- **CORE_PARITY.C**—This program demonstrates the use of parity modes by repeatedly sending byte values 0–127 from Serial Port B to Serial Port C. The program will switch between generating parity or not on Serial Port B. Serial Port C will always be checking parity, so parity errors should occur during every other sequence.

  To set up the Prototyping Board, you will need to tie PC4 and PC3 (TxB and RxC) together at header J4 as shown in the diagram.

  The Dynamic C **STDIO** window will display the error sequence.

3.4 Upgrading Dynamic C

Dynamic C patches that focus on bug fixes are available from time to time. Check the Web sites

- www.zworld.com/support/

or

- www.rabbitsemiconductor.com/support/

for the latest patches, workarounds, and bug fixes.

3.4.1 Add-On Modules

Dynamic C installations are designed for use with the board they are included with, and are included at no charge as part of our low-cost kits. Z-World offers add-on Dynamic C modules for purchase, including the popular μC/OS-II real-time operating system, as well as PPP, Advanced Encryption Standard (AES), and other select libraries.

In addition to the Web-based technical support included at no extra charge, a one-year telephone-based technical support module is also available for purchase.

NOTICE TO USERS

Z-WORLD PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE-SUPPORT DEVICES OR SYSTEMS UNLESS A SPECIFIC WRITTEN AGREEMENT REGARDING SUCH INTENDED USE IS ENTERED INTO BETWEEN THE CUSTOMER AND Z-WORLD PRIOR TO USE. Life-support devices or systems are devices or systems intended for surgical implantation into the body or to sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling and user's manual, can be reasonably expected to result in significant injury.

No complex software or hardware system is perfect. Bugs are always present in a system of any size. In order to prevent danger to life or property, it is the responsibility of the system designer to incorporate redundant protective mechanisms appropriate to the risk involved.

All Z-World products are 100 percent functionally tested. Additional testing may include visual quality control inspections or mechanical defects analyzer inspections. Specifications are based on characterization of tested sample units rather than testing over temperature and voltage of each unit. Z-World products may qualify components to operate within a range of parameters that is different from the manufacturer's recommended range. This strategy is believed to be more economical and effective. Additional testing or burn-in of an individual unit is available by special arrangement.

INDEX

A

additional information
  online documentation 5
  references 4

D

description 1
Development Kit 7
Dynamic C 3
  add-on modules 29
  features 15, 21
  installing 17
  multitasking 22
  sample programs 18
    break point 20
    editing a program 21
    single-stepping 20
    watch expressions 20
  standard features 16
    debugging 16
  telephone-based technical support 29
  upgrades and patches 29
  USB port settings 12

F

features
  Prototyping Board 8
  RCM2000 1

H

hardware connections 9
  install RCM2000 on Prototyping Board 9
  power supply 11
  programming cable 10
hardware reset 11

M

models
  factory versions 1

P

pinout
  RCM2000 2
power supply
  connections 11
programming cable
  RCM2000 connections 10
Prototyping Board 8
  features 8
  mounting RCM2000 9

R

RCM2000
  mounting on Prototyping Board 9
reset 11

S

sample programs 18
  FLASHLED.C 19
  getting to know the RCM2000
    EXTSRAM.C 25
    FLASHLED.C 25
    FLASHLED2.C 26
    FLASHLEDS.C 26
    FLASHLEDS2.C 26
    KEYLCD.C 26
    LCD_DEMO.C 27
    SWTEST.C 27
    TOGGLELED.C 27
  PONG.C 12
  serial communication
    CORE_FLOWCONTROL.C 28
    CORE_PARITY.C 28
software
  sample programs 18
specifications
  physical and electrical 2

T

technical support 13

U

USB/serial port converter 10
  Dynamic C settings 12

SCHEMATICS

090-0097 RCM2000 Schematic

www.rabbitsemiconductor.com/documentation/schemat/090-0097.pdf

090-0099 RCM2000 Prototyping Board Schematic

www.rabbitsemiconductor.com/documentation/schemat/090-0099.pdf

090-0128 Programming Cable Schematic

www.rabbitsemiconductor.com/documentation/schemat/090-0128.pdf

The schematics included with the printed manual were the latest revisions available at the time the manual was last revised. The online versions of the manual contain links to the latest revised schematic on the Web site. You may also use the URL information provided above to access the latest schematics directly.